

日本国特許庁
JAPAN PATENT OFFICE

U3-0155-SM

12

JC997 U.S. PTO
09/935686



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日

Date of Application:

2000年 8月28日

出願番号

Application Number:

特願2000-257184

出願人

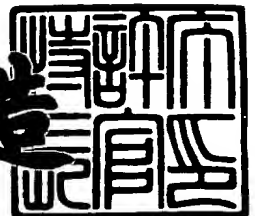
Applicant(s):

株式会社デンソー

2001年 7月19日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造



出証番号 出証特2001-3064442

【書類名】 特許願

【整理番号】 N000645

【提出日】 平成12年 8月28日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/00

【発明者】

 【住所又は居所】 愛知県刈谷市昭和町1丁目1番地 株式会社デンソー内

 【氏名】 神谷 政裕

【発明者】

 【住所又は居所】 愛知県刈谷市昭和町1丁目1番地 株式会社デンソー内

 【氏名】 江川 邦隆

【発明者】

 【住所又は居所】 愛知県刈谷市昭和町1丁目1番地 株式会社デンソー内

 【氏名】 松田 啓資

【発明者】

 【住所又は居所】 愛知県刈谷市昭和町1丁目1番地 株式会社デンソー内

 【氏名】 石原 秀昭

【特許出願人】

 【識別番号】 000004260

 【氏名又は名称】 株式会社デンソー

【代理人】

 【識別番号】 100071135

 【住所又は居所】 名古屋市中区栄四丁目6番15号 名古屋あおば生命ビル

 【弁理士】

 【氏名又は名称】 佐藤 強

 【電話番号】 052-251-2707

【手数料の表示】

 【予納台帳番号】 008925

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9200169

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 コンパイラ、記録媒体、プログラム変換装置、プログラム変換方法及びマイクロコンピュータ

【特許請求の範囲】

【請求項1】 プログラムとして記述されたソースコードからメモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するRISC型CPUのオブジェクトコードを生成するコンパイラにおいて、

前記ソースコード中にビット演算式が記述されている場合は、そのビット演算式を演算対象である各ビット変数の値に基づく条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成することを特徴とするコンパイラ。

【請求項2】 複数のビット変数の集合をレジスタ変数として、前記RISC型CPUに内蔵されるレジスタに割り当てることが可能であることを特徴とする請求項1記載のコンパイラ。

【請求項3】 請求項1または2記載のコンパイラプログラムが記憶されていることを特徴とする記録媒体。

【請求項4】 プログラムとして記述されたソースコードファイルの内容を読み出し、メモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するRISC型CPUのオブジェクトコードファイルを生成するプログラム変換装置において、

前記ソースコード中にビット演算式が記述されている場合は、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成するコンパイラプログラムが記憶された記憶手段を備えたことを特徴とするプログラム変換装置。

【請求項5】 前記コンパイラは、複数のビット変数の集合をレジスタ変数

として、前記RISC型CPUに内蔵されるレジスタに割り当てることが可能であることを特徴とする請求項4記載のプログラム変換装置。

【請求項6】 プログラムとして記述されたソースコードファイルの内容を読み出し、メモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するRISC型CPUのオブジェクトコードファイルを生成するプログラム変換方法において、

前記ソースコード中にビット演算式が記述されている場合に、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成することを特徴とするプログラム変換方法。

【請求項7】 複数のビット変数の集合をレジスタ変数として、前記RISC型CPUに内蔵されるレジスタに割り当ててことを特徴とする請求項6記載のプログラム変換方法。

【請求項8】 メモリ若しくはレジスタ上に配置されるビット変数の値を判定する命令と、前記ビット変数に対して即値を代入する命令とを有するRISC型CPUと、

プログラムとして記述されたソースコード中に記述されているビット演算式について、その演算対象である各ビット変数の値を夫々判定し、各判定の結果に基づいて前記ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成するコンパイラによってコード変換されたプログラムが記憶されているプログラムメモリとを備えたことを特徴とするマイクロコンピュータ。

【請求項9】 前記プログラムメモリに記憶されているプログラムは、ソースコード中に記述されている複数のビット変数の集合をレジスタ変数として、前記RISC型CPUに内蔵されるレジスタに割り当てられるように、前記コンパイラによってオブジェクトコードが生成されていることを特徴とする請求項8記載のマイクロコンピュータ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、プログラムとして記述されたソースコードからRISC型CPUのオブジェクトコードを生成するコンパイラ、そのコンパイラプログラムが記憶された記録媒体、プログラム変換装置、プログラム変換方法及びマイクロコンピュータに関する。

【0002】

【従来の技術】

現在、CPUやマイクロコンピュータなどのプログラム開発は、例えばC言語やC++言語などの高級言語を用いて行われる場合が一般的である。高級言語によって記述されたソースコードは、コンパイラによってプログラム変換（コンパイル）され、CPU等が直接実行可能な命令形式（機械語）のオブジェクトコードが生成されるようになっている。コンパイラによって生成されたオブジェクトコードは、プログラムメモリに格納され、CPU等によって読み出されることで所期の処理が実行されるようになっている。

【0003】

ところで、RISC(Reduced Instruction Set Computer)型CPUは、機能が単純で実行時間が短い命令だけを使用可能とし、複数の命令の実行に伴うプロセスにおける各ステージ（フェッチ、デコード、実行、書込みなど）を複数の命令パイプラインによって並行に実行することでプログラム実行速度の高速化を図ったものである。そのため、ソースコードからコンパイラによって生成されるオブジェクトコードの容量は、CISC(Complex Instruction Set Computer)型CPUに比較して多くなる傾向を示す。

【0004】

図7は、従来のコンパイラによって生成されるオブジェクトコード（ニーモニックで表現）の例を示すプログラムリストである。ソースコードは、

$$\text{bit1} = \text{bit2} \& \text{bit3} \quad \dots (1)$$

であるとする。即ち、ビット変数bit2と、ビット変数bit3との論理積をとった結果をビット変数bit1に代入する、というビット演算である。また、

各ビット変数 `bit1`～`bit3` は、8ビット変数 `BIT1`～`BIT3` の第4ビット（LSBから第3ビット目）に配置されている変数であるとする。

また、図8は、(1)式のビット演算のソースコードをコンパイラが変換する場合のプロセスを示すフローチャートである。

【0005】

図7において、オブジェクトコードの①部分では、例えばメモリ上に配置された8ビット変数 `BIT2` の内容を汎用レジスタ `reg1` に読み出してから `reg1` を3ビット右シフトして、その第4ビットをLSBに位置させる。そして、レジスタ `reg1` とマスクデータ `$0001`（“\$”は16進数を示すシンボル）との論理積をとることで、レジスタ `reg1` のLSB以外のビットをクリアする。ここまでが、図8のステップA1に対応する。オブジェクトコードの②部分では8ビット変数 `BIT3` について同様の処理を行っており、図8のステップA2に対応する。

【0006】

オブジェクトコード③部分では、レジスタ `reg1`、`reg2` の論理積演算を行う（図8のステップA3に対応）。続くオブジェクトコード④部分では、前記論理積演算の結果が格納されているレジスタ `reg1` を3ビット左シフトさせると、8ビット変数 `BIT1` の内容をレジスタ `reg2` に読み出し、レジスタ `reg2` とマスクデータ `$fff7` との論理積演算を行うことで、代入先となるレジスタ `reg2` の第4ビットをクリアし、それ以外のビットは保存する。そして、レジスタ `reg1`、`reg2` の論理和演算を行い、その論理積演算結果が格納されているレジスタ `reg2` の内容をメモリ上の8ビット変数 `BIT1` に転送すると（図8のステップA4に対応）、一連の演算処理が終了する。

【0007】

【発明が解決しようとする課題】

このように、RISC型CPUでは命令数が少なくビット変数の値を読み出すための専用命令がないので、他の複数の命令の組み合わせによって所期の処理を実行するようにコード変換されている。その結果、ソースコードにおいて1ステップで記述されたビット演算を実行するのに、オブジェクトコードでは18ステ

ップを要している。従って、CPUのプログラム記憶領域をより多く必要とすることになり、また、ステップ数が多くなった分だけ処理速度が低下するという問題があった。

【0008】

本発明は上記事情に鑑みてなされたものであり、その目的は、プログラム容量をより小さくでき、且つ、処理速度を高速化できるようにオブジェクトコードを生成するコンパイラ、そのコンパイラプログラムが記憶された記録媒体、プログラム変換装置、プログラム変換方法及びマイクロコンピュータを提供することにある。

【0009】

【課題を解決するための手段】

請求項1記載のRISC型CPUのコンパイラによれば、プログラムとして記述されたソースコード中にビット演算式が記述されている場合は、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定する式に変換する。そして、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードが生成される。

【0010】

例えば、ビット演算が、ビット変数aとビット変数bとの論理積演算を行った結果をビット変数cに代入するというビット演算、即ち、

$$c = a \& b \quad \dots (2)$$

であるとする、(2)式右辺に記述されたビット演算式“a & b”の結果が“1”となるのは“a = 1”且つ“b = 1”の場合であり、例えばC言語においてif文やwhile文などに使用される条件判定式で表現すれば、

$$a == 1 \ \&\& \ b == 1 \quad \dots (3)$$

となる。つまり、(2)式は、条件判定式(3)が真である場合に(2)式左辺のビット変数cに“1”を代入し、条件判定式(3)が偽である場合にビット変数cに“0”を代入することと等価であり、ビット演算式は条件判定式に置き換えることが可能である。

【0011】

従って、ソースコード中に記述されているビット演算式については、置き換えた判定式に基づいて、各ビット変数の値を判定し、各判定の結果に基づいて条件分岐を行うなどして、前記ビット演算の結果が格納されるビット変数に異なる即値“0, 1”を代入するようにオブジェクトコードを生成すれば、ビット演算と等価な処理を行うことができる。

【0012】

即ち、ビット演算について斯様にオブジェクトコードを生成することで、RISC型CPUが実行する命令のステップ数が従来よりも短くなる。そして、一般に、CPUのプログラム中においてビット演算を行う頻度は比較的高いことから、プログラム全体として生成されるオブジェクトコードの量を少なくすることができる。そして、プログラムの容量（サイズ）を従来よりも小さくすることができると共に、RISC型CPUがビット演算を実行する場合の処理速度を高速化することが可能となる。

【0013】

請求項2記載のコンパイラによれば、所謂ビットフィールドと称する複数のビット変数の集合をレジスタ変数としてRISC型CPUに内蔵されるレジスタに割り当てることができるので、ユーザはプログラミングを多様に行うことができ、RISC型CPUによる処理を効率的に行わせることが可能となる。

【0014】

請求項4記載のRISC型CPUのプログラム変換装置によれば、記憶手段に記憶されているコンパイラは、プログラムとして記述されたソースコード中にビット演算式が記述されている場合は、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定式に変換する。そして、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードが生成される。

【0015】

即ち、論理積や論理和などのビット演算は、上述のように、各ビット変数の値

に基づいて条件判定を行う式に置き換えることが可能である。従って、ソースコード中に記述されているビット演算式については、置き換えた条件判定式に基づいて、各ビット変数の値を判定し、各判定の結果に基づいて条件分岐を行うなどして、前記ビット演算の結果が格納されるビット変数に異なる即値“0, 1”を代入するようにオブジェクトコードを生成すれば、ビット演算と等価な処理を行うことができる。

【0016】

そして、ビット演算について斯様にオブジェクトコードを生成することで、RISC型CPUが実行する命令のステップ数が従来よりも短くなる。一般に、CPUのプログラム中においてビット演算を行う頻度は比較的高いことから、プログラム全体として生成されるオブジェクトコードの量が少なくなり、変換されたプログラムの容量を従来よりも小さくすることができる。また、RISC型CPUがビット演算を実行する場合の処理速度が高速化される。

【0017】

請求項5記載のプログラム変換装置によれば、所謂ビットフィールドと称する複数のビット変数の集合をレジスタ変数としてRISC型CPUに内蔵されるレジスタに割り当てるので、ユーザはプログラミングを多様に行うことができ、RISC型CPUによる処理を効率的に行わせることが可能となる。

【0018】

請求項8記載のマイクロコンピュータによれば、RISC型CPUが実行するプログラムが記憶されているプログラムメモリには、ソースコード中に記述されているビット演算式については、そのビット演算式を、演算対象である各ビット変数の値に基づいて条件を判定式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数に異なる即値を夫々代入する処理を行うようにオブジェクトコードを生成するコンパイラによってコード変換されたプログラムが記憶されている。

【0019】

即ち、論理積や論理和などのビット演算は、上述のように、各ビット変数の値に基づいて条件判定を行う式に置き換えることが可能である。従って、ソースコ

ード中に記述されているビット演算式については、置き換えた条件判定式に基づいて、各ビット変数の値を判定し、各判定の結果に基づいて条件分岐を行うなどして、前記ビット演算の結果が格納されるビット変数に異なる即値“0, 1”を代入するようにオブジェクトコードを生成すれば、ビット演算と等価な処理を行うことができる。

【0020】

そして、ビット演算について斯様にオブジェクトコードを生成することで、RISC型CPUが実行する命令のステップ数が従来よりも短くなるので、プログラム全体として生成されるオブジェクトコードの量が少なくなり、プログラムメモリの容量を従来よりも小さくすることができる。また、RISC型CPUがビット演算を実行する場合の処理速度を高速化することが可能となる。

【0021】

請求項9記載のマイクロコンピュータによれば、プログラムメモリに記憶されているプログラムは、ソースコード中に記述されている複数のビット変数（所謂ビットフィールド）をレジスタ変数として、RISC型CPUに内蔵されるレジスタに割り当てられるようにコンパイラによってオブジェクトコードが生成されるので、RISC型CPUによる処理を効率的に行わせることが可能となる。

【0022】

【発明の実施の形態】

以下、本発明の一実施例について図1乃至図6を参照して説明する。図4は、プログラム変換装置の構成を示す図である。プログラム変換装置たるパーソナルコンピュータ（パソコン）若しくはワークステーション1には、コンパイラ2がインストールされている。具体的には、パソコン1の本体1aに内蔵されているハードディスク等の記憶装置（記憶手段）にコンパイラ2のプログラムファイルが記憶されている。

【0023】

本体1a内蔵の記憶装置には、図5に示すように、ユーザが例えばC言語などの高級言語によって記述したソースコードファイル3も記憶されている。そして、ユーザは、コンパイラ2のプログラムをパソコン1上で起動することで、ソー

スコードファイル3からオブジェクトコードファイル4を変換生成させる。コンパイラ2は、ソースコードファイル3に記述されているC言語のソースコードを読み出してその内容を解釈すると、CPU等が固有の命令によって対応する処理を最も効率的に実行できるようなオブジェクトコードを生成するようにコンパイルを行う。

【0024】

パソコン 1 の本体 1 a には ROM ライタ 5 が接続されており、両者は、例えば RS-232C 等のシリアル通信プロトコルによって通信が可能となっている。コンパイラ 2 によって生成されたオブジェクトコードファイル 4 は ROM ライタ 5 に転送されると、ROM ライタ 5 にセットされたマイクロコンピュータ（マイコン）6 に内蔵されているプログラムメモリ 7（図 6 参照）にバイナリデータとして書き込まれるようになっている。

【0025】

図6は、マイコン6の電氣的構成を示す機能ブロック図である。マイコン6は、RISC型のCPU8、EEPROMやフラッシュROMなどからなるプログラムメモリ7及びSRAMなどからなる内部メモリ9を備え、ワンチップマイコンとして構成されている。また、マイコン6の外部には、DRAMなどからなる外部メモリ10が接続されている。CPU8と、プログラムメモリ7、内部メモリ9及び外部メモリ10は、アドレスバス11及びデータバス12を介して接続されている。

【 0 0 2 6 】

ＣＰＵ ８の内部には、演算等を行うための演算部（ＡＬＵ（Alithmetic Logical Unit））１３，演算部１３が演算等を行う場合に使用される複数のレジスタが配置されているレジスタ部１４，及びレジスタ部１４等にロード／ストア等の制御を行うための制御部１５などが配置されている。

【 0 0 2 7 】

また、CPU8は、前提構成として固有の命令に、少なくとも、

ビットテスト命令 : `test (. b/w/l)`

ビットセット命令 : set (. b/w/l)

ビットクリア命令 : `clr (. b/w/1)`

を備えている。

【0028】

ビットテスト命令は、指定されたビット変数の値をテスト（判定）する命令であり、そのテスト結果“0，1”は、CPU8のレジスタ部14内にあるコンディションコードレジスタ（CCR）のZフラグに制御部15によって反映されるようになっている。即ち、テスト結果が“0”であれば、CCRのZフラグがセットされ、テスト結果が“1”であればZフラグはクリアされる。

【0029】

また、このビットテスト命令は、ユーザの定義などによりビット変数がメモリに割り当てられている場合と、レジスタ部14内の汎用レジスタに割り当てられている場合との何れであっても実行することが可能である。`(. b/w/1)`は、メモリ上のビット変数をテスト等する場合のアクセスサイズを指定するものであり、（バイト（8）／ワード（16）／ロングワード（32））に対応する。尚、レジスタ上のビット変数をテスト等する場合は、アクセスサイズの指定は不要である。

また、ビットセット命令は、指定されたビット変数に即値“1”を代入する命令であり、ビットクリア命令は、指定されたビット変数に即値“0”を代入する命令である。

【0030】

次に、本実施例の作用について図1乃至図3をも参照して説明する。図1は、コンパイラ2がビット演算に関するコンパイル処理を行う場合のプロセスを示すフローチャートである。先ず、コンパイラ2は、C言語で記述された右辺のビット演算式を解釈（デコード）すると、ブール代数規則に基づき結果が1（真）または0（偽）となる条件判定式に変換する。そして、その条件判定式に対応するオブジェクトコードを出力する（ステップS1）。

【0031】

次に、コンパイラ2は、条件判定式が真であった場合に、ビット演算式の結果が代入される左辺のビット変数に“1”若しくは“0”（異なる即値）を代入す

るためのコードを生成し、それから、処理終了へ分岐するためのコードを生成する（ステップ S 2）。それから、条件判定式が偽であった場合に、ビット演算式の結果が代入される左辺のビット変数に“0”若しくは“1”を代入するためのコードを生成すると（ステップ S 3）と、処理終了となる。

【0032】

ここで、図 2 は、図 1 のフローチャートに基づいて、具体的に（1）式のビット演算をコンパイル処理してオブジェクトコードを生成する場合のフローチャートであり、図 3 は、図 2 のフローチャートに従うコンパイル処理によって生成されるオブジェクトコードをニーモニックで示すプログラムリストである。

【0033】

（1）式右辺に記述されたビット演算式“`bit 2 & bit 3`”の結果が“1”となるのは、“`bit 2 = 1`”且つ“`bit 3 = 1`”の場合であり、C 言語における条件判定式で表現すると、

$$\text{bit 2} == 1 \quad \&\& \quad \text{bit 3} == 1 \quad \dots (4)$$

となる。従って、（1）式は、条件判定式（4）が真である場合に（1）式左辺のビット変数 `bit 1` に“1”を代入し、条件判定式（4）が偽である場合にビット変数 `bit 1` に“0”を代入することと等価である。

【0034】

そこで、図 2 においては、先ずビット変数 `bit 2` の値を判定し（ステップ B 1）、その値が“0”であればビット変数 `bit 1` に“0”を書き込んで（ステップ B 3）処理を終了する。また、ビット変数 `bit 2` の値が“1”である場合はビット変数 `bit 1` の値を判定し（ステップ B 2）、その値が“0”であればステップ B 3 に移行し、ビット変数 `bit 1` の値が“1”であればビット変数 `bit 1` に“1”を書き込んで（ステップ B 4）処理を終了する。

【0035】

即ち、図 2 のフローチャートでは、ステップ B 1、B 2 が図 1 のステップ S 1 に基づく処理であり、ステップ B 1、B 2 の何れかよりステップ B 3 に移行するケースが条件判定式（4）が偽である場合に対応する。そして、ステップ B 3 は、図 1 のステップ S 3 に基づく処理である。また、ステップ B 2 よりステップ B

4に移行するケースが条件判定式(4)が真である場合に対応し、ステップB4は、図1のステップS2に基づく処理である。

【0036】

次に、実際のコンパイル処理結果のプログラムリストである図3を参照すると、ステップ①でビット変数bit2の値を判定し、ステップ②ではその値が“0”であればラベルLOO1に分岐させる(ステップB1に対応)。例えば、ビット変数bit2の値が“0”であれば、前述したように、CPU8のレジスタ部14内にあるCCRのZフラグがセットされ、ステップ②での条件判定は真となる。

【0037】

次のステップ③、④では、ビット変数bit3に関してステップ①、②と同様の処理を行う(ステップB2に対応)。以上のステップ①～④が図1のステップS1を実行した結果であり、条件判定式(4)に対応して出力されたオブジェクトコードとなる。そして、ステップ②、④の何れかにおいてラベルLOO1に分岐するケースが、条件判定式の結果が偽の場合に対応する。また、ステップ②、④の何れにおいてもラベルLOO1に分岐することなく、次のステップ⑤を実行するケースが、条件判定式の結果が真の場合に対応する。

【0038】

条件判定式の結果が真の場合に対応するステップ⑤では、ビット変数bit1に“1”をセットし、このビット演算処理を終了するためにラベルLOO2に分岐する(ステップ⑥)。この部分が図1のステップS2を実行した結果であり、図2ではステップB4に対応する。

また、条件判定式の結果が偽の場合に対応するステップ⑦では、ビット1をゼロクリアして処理を終了する。この部分が図1のステップS3を実行した結果であり、図2ではステップB3に対応する。

【0039】

斯様にコンパイラ2によって生成されたオブジェクトコードのサイズは、図7に示す従来のコンパイラによって生成されたコードのサイズに比較すると大幅に縮小されている。

【0040】

以上のようにコンパイラ2によってコンパイル処理が行われた結果生成されたオブジェクトコードファイル4は、上述のようにパソコン1よりROMライタ5に転送され、ROMライタ5にセットされたマイコン6内蔵のプログラムメモリ7にバイナリデータとして書き込まれる。

【0041】

そして、マイコン6のCPU8は、プログラムメモリ7に記憶されたオブジェクトコードをプログラムとして読み出して実行する。尚、ビット変数bit1～bit3は、ユーザの定義により内部メモリ9、外部メモリ10またはレジスタ部14内の汎用レジスタなどの何れに配置されているものでも構わない。ビット変数が汎用レジスタ上に割り付けられている場合は、例えば、図3のステップ①における“tst. b bit2”に代えて、“tst bit2”がオブジェクトコードとして生成される。

【0042】

ところで、例えば、ビット変数bit2が内部メモリ9上に配置されている場合、図3のステップ①における“tst. b bit2”を実行すると、CPU8の演算部13は、内部メモリ9よりビット変数bit2の内容を読み出して判定を行う。この場合、例えば、図6に示すコードの“mov BIT2, reg1”などの命令が1クロック（CPU8の動作クロック）で実行可能であるとすると、“tst. b bit2”の実行には2～3クロックを要することになるが、コードサイズの全体が大幅に縮小されているので、トータルでの処理時間は短縮される。

【0043】

以上のように本実施例によれば、パソコン1の本体1aにインストールされたコンパイラ2は、プログラムとして記述されたソースコードファイル3のソースコード中に記述されているビット演算式を、演算対象である各ビット変数bit2, bit3の値に基づいて条件を判定する式に変換し、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数bit1に異なる即値“0, 1”を夫々代入する処理を行うようにオ

プロジェクトコードを生成し、RISC型CPU8に最適なオブジェクトコードファイル4を出力するようにした。

【0044】

従って、CPU8が実行する命令のステップ数が従来よりも短くなり、プログラム全体として生成されるオブジェクトコードの量が少なくなるので、プログラムの容量を従来よりも小さくすることができると共に、CPU8がビット演算を実行する場合の処理速度を高速化することが可能となる。

【0045】

また、マイコン6のプログラムメモリ7には、コンパイラ2によってコード変換されたプログラムが記憶され、マイコン6のCPU8は、プログラムメモリ7に記憶されたオブジェクトコードをプログラムとして読み出して実行するので、CPU8が実行する命令のステップ数が従来よりも短くなって、プログラム全体として生成されるオブジェクトコードの量が少なくなり、プログラムメモリ7の容量を従来よりも小さくすることができる。そして、CPU8がビット演算を実行する場合の処理速度が高速化されるので、マイコン6の処理能力を向上させることができる。

【0046】

本発明は上記し且つ図面に記載した実施例にのみ限定されるものではなく、次のような変形または拡張が可能である。

ビット演算式は、論理積に限ることなく、論理和や排他的論理和などを行うものでも良い。その場合、論理の違いに応じて、条件分岐コマンド“beq”に代えて“bne(not equal)”などをオブジェクトコードとして生成すれば良い。

また、ソースコードにおいて複数のビット変数の集合、所謂ビットフィールドがレジスタ変数としてCPU8のレジスタ部14に割り当てている場合であっても、コンパイラ2は、ビットテスト命令によってビットフィールドの各ビット変数毎に値の判定を行うようにコードを出力する。従って、ユーザはプログラミングを多様に行うことができ、CPU8による処理を効率的に行わせることが可能となる。

【0047】

プログラムメモリ7に対するプログラムデータの書き込みは、ROMライタ5を使用するものに限らない。例えば、マイコン6側にRS-232C等のシリアル通信インターフェイスを設けて、パソコン1側よりマイコン6にプログラムデータを直接送信することで、CPU8によってプログラムデータをプログラムメモリ7に書き込ませるようにしても良い。

コンパイラ2のプログラムを、CD-ROM、フロッピーディスクやMOディスク等の記録媒体に記憶させても良く、また、そのような記録媒体に記憶されている状態で、パソコン1の主記憶上にプログラムを読み出してコンパイル処理を行っても良い。

マイクロコンピュータとしては、マイコン6のようなワンチップマイコンに限ることなく、CPU8とプログラムメモリ7とが夫々独立のチップとして形成されて同一のプリント基板上に搭載されているようなマイクロコンピュータであっても良い。この場合、ROMライタ5には、プログラムメモリ7のみを接続してプログラムデータを書き込むようにすれば良い。

また、プログラムメモリは、書き替え可能なメモリに限らず、例えばマスクROMなどの書き替え不可能なメモリを用いても良い。

【図面の簡単な説明】

【図1】

本発明の一実施例であり、コンパイラがビット演算に関するコンパイル処理を行う場合のプロセスを示すフローチャート

【図2】

図1のフローチャートに基づいて、コンパイラが(1)式のビット演算をコンパイル処理してオブジェクトコードを生成する場合のフローチャート

【図3】

図2のフローチャートに従うコンパイル処理によって生成されるオブジェクトコードをニーモニックで示すプログラムリスト

【図4】

プログラム変換装置たるパーソナルコンピュータ若しくはワークステーション

の構成を示す図

【図 5】

コンパイラが、ソースコードファイルからオブジェクトコードファイルを変換生成する処理を概念的に示す図

【図 6】

マイクロコンピュータの電氣的構成を示す機能ブロック図

【図 7】

従来技術を示す図 3 相当図

【図 8】

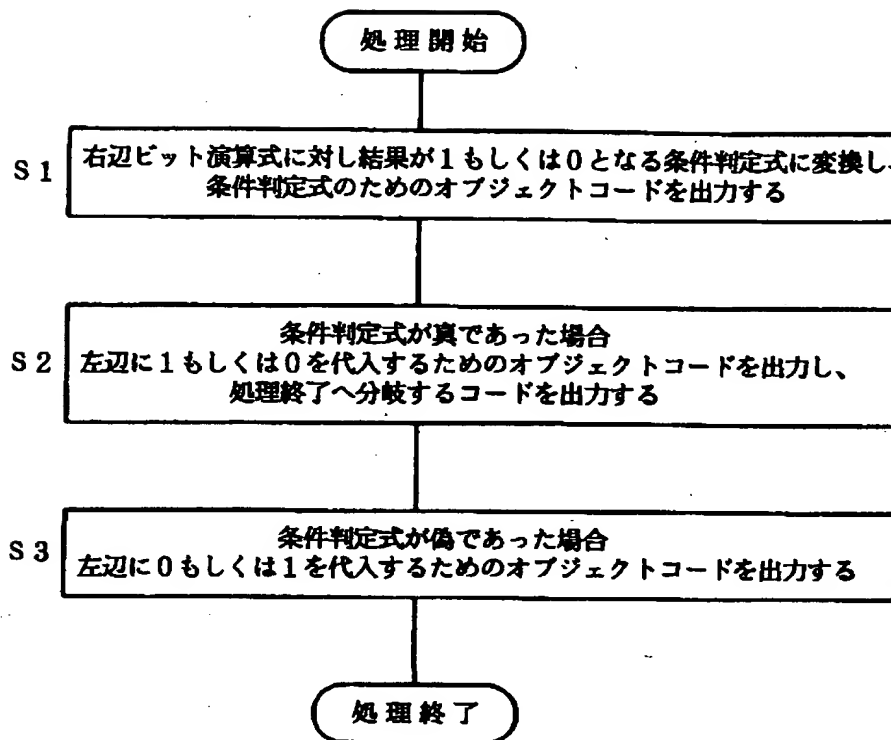
(1) 式のビット演算のソースコードを、従来のコンパイラが変換する場合のプロセスを示すフローチャート

【符号の説明】

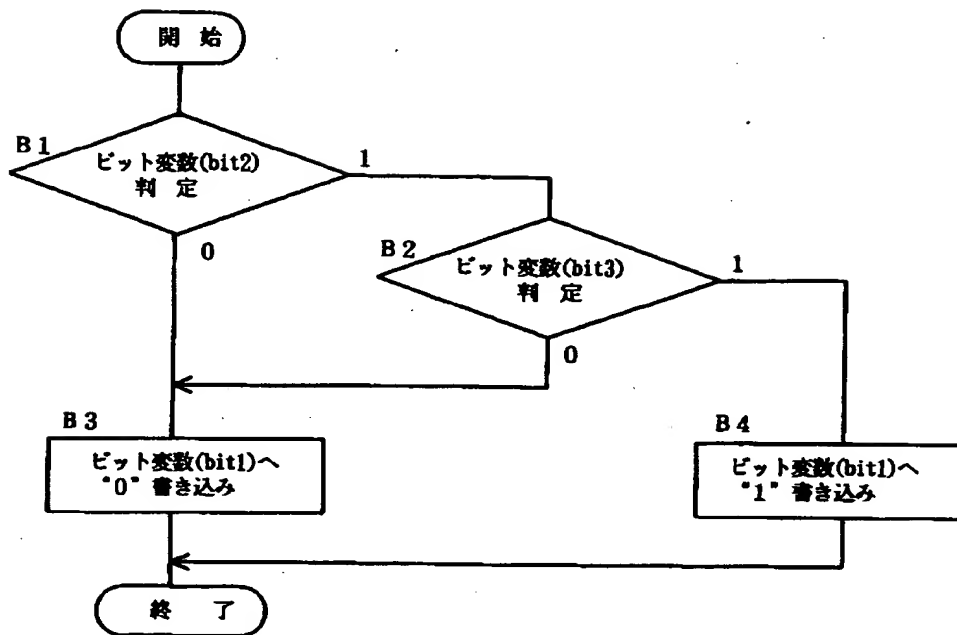
1 はパーソナルコンピュータ若しくはワークステーション（プログラム変換装置）、2 はコンパイラ、3 はソースコードファイル、4 はオブジェクトコードファイル、6 はマイクロコンピュータ、7 はプログラムメモリ、8 は CPU を示す。

【書類名】 図面

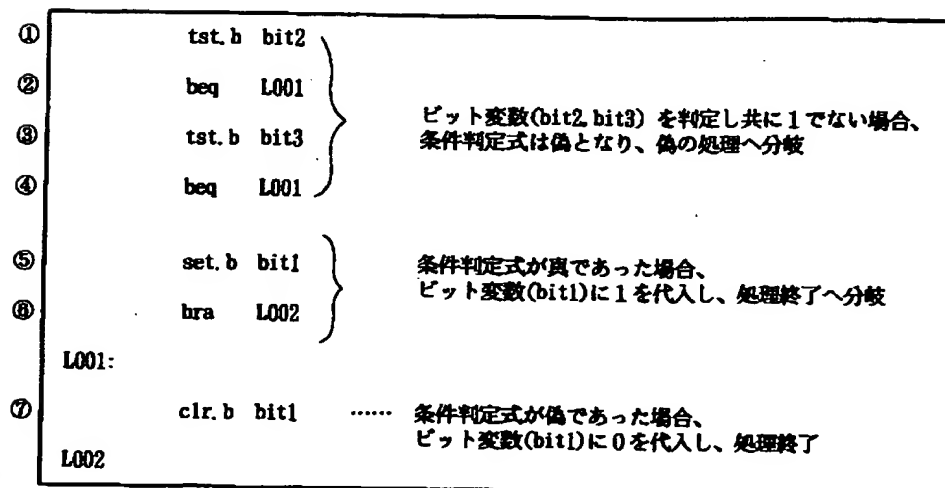
【図 1】



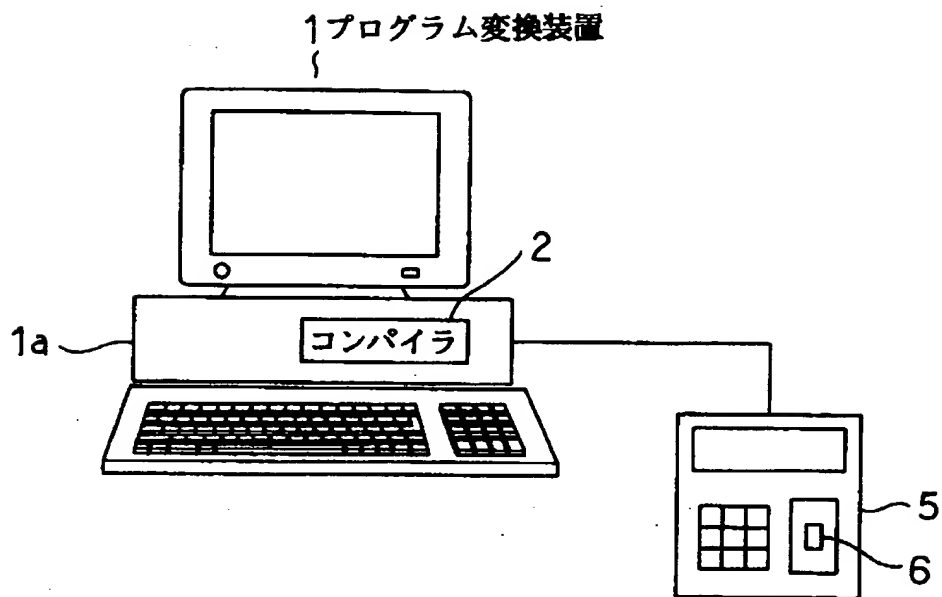
【図 2】



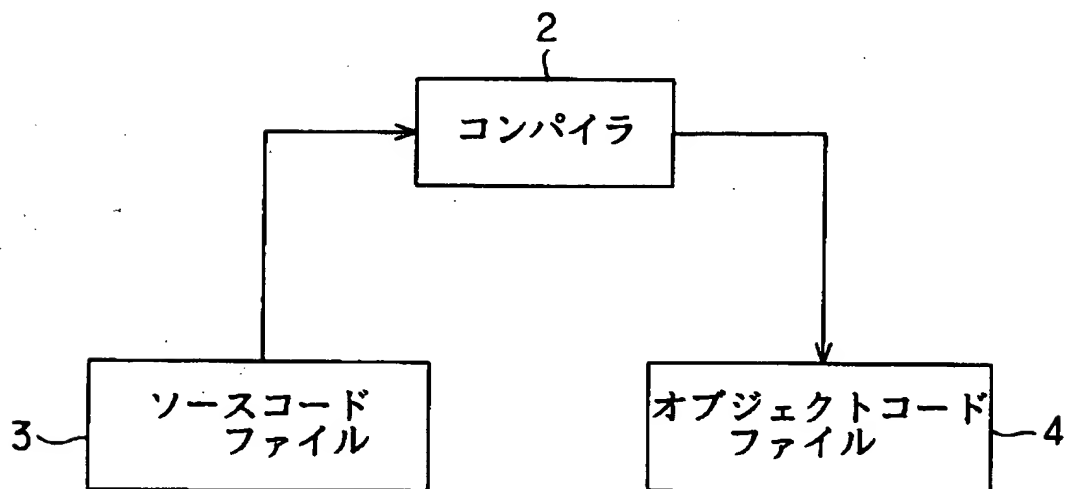
【図 3】



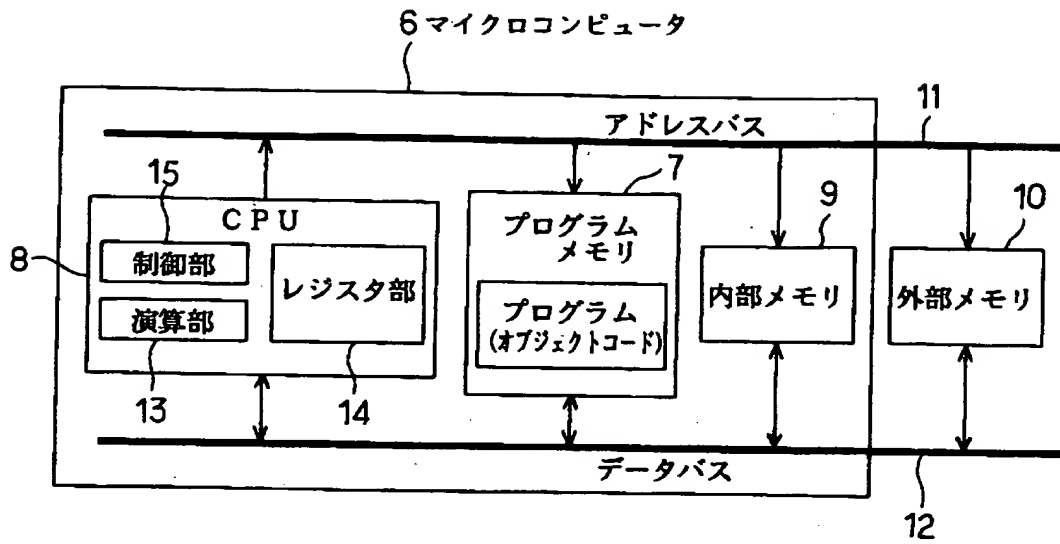
【図4】



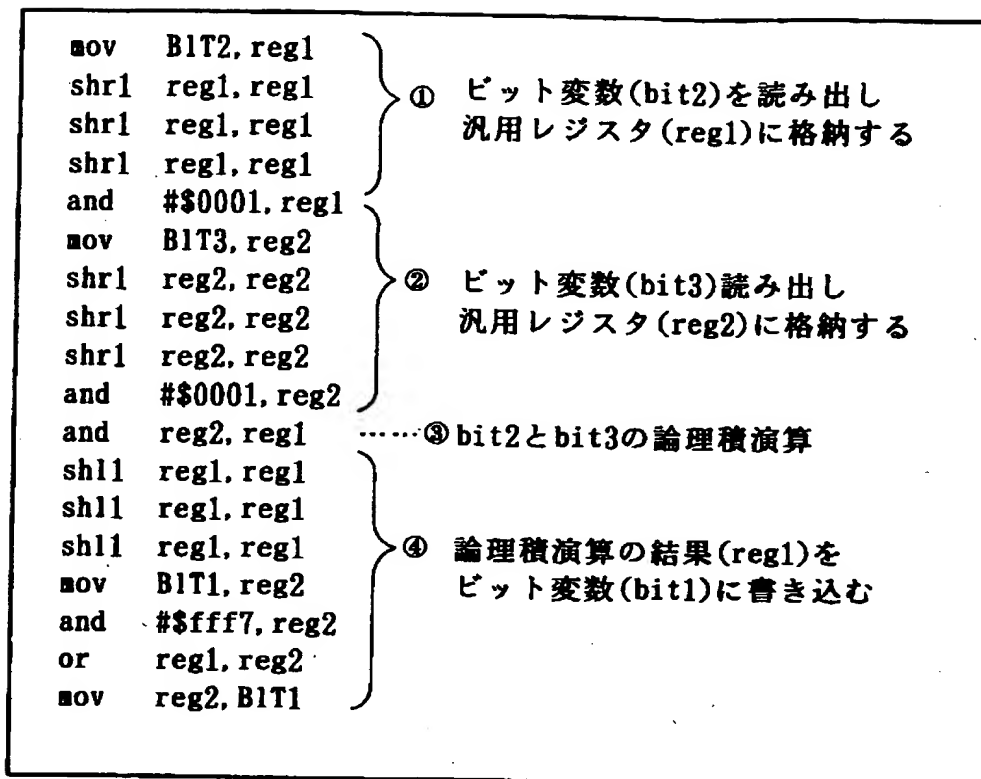
【図5】



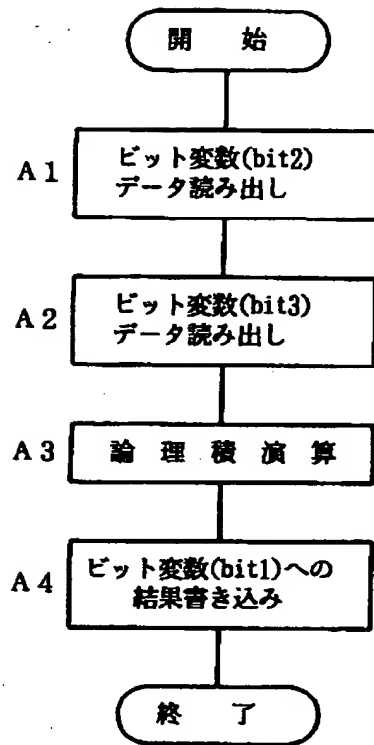
【図6】



【図 7】



【図 8】



【書類名】 要約書

【要約】

【課題】 プログラム容量をより小さくでき、且つ、処理速度を高速化できるようにオブジェクトコードを生成するコンパイラを提供する。

【解決手段】 コンパイラは、プログラムとして記述されたソースコードファイルのソースコード中に記述されているビット演算式を、演算対象である各ビット変数 `bit 2`、`bit 3` の値に基づいて条件を判定する式に変換し（ステップ `S 1`）、その判定式の結果が真である場合と偽である場合との夫々に対応して、ビット演算の結果が格納されるビット変数 `bit 1` に異なる即値 “0, 1” を夫々代入する処理を行うようにオブジェクトコードを生成し（ステップ `S 2`, `S 3`）、RISC型CPUに最適なオブジェクトコードファイルを出力する。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [000004260]

| | |
|----------|-----------------|
| 1. 変更年月日 | 1996年10月 8日 |
| [変更理由] | 名称変更 |
| 住 所 | 愛知県刈谷市昭和町1丁目1番地 |
| 氏 名 | 株式会社デンソー |